

CS 113 – Computer Science I

Lecture 16 – Sorting

Adam Poliak

11/03/2022

Announcements

- Assignment 08
 - Due tonight Thursday 11/10

Guessing game – in class exercise

Pair up:

- Person A chooses a number between 1 and 64
- Person B guesses the number
- Until the guess is correct:
 - Person A tells whether the guess is too high or too low
 - Person B guesses again

After 2 rounds each, choose a number between 1 and 512

Binary Search run time

As the size of our collection increases, the number of guesses/comparisons increases, but not *linearly*

The time increases by $\log n$ (we use base 2)

If our collection contains 8 data points, how many comparisons in worst case do we make:

$$\log_2 8 = 3$$

If our collection contains 512 data points, how many comparisons in worst case do we make:

$$\log_2 512 = 9$$

Sorting

Bubble Sort

Compare two adjacent items, and swap if needed

Repeat until largest item is at the back

Repeat process until done

Bubble Sort

0	1	2	3	4	5
10	4	3	0	11	8

What do we do first?

Bubble Sort

len = 6



↑
j - 1
0

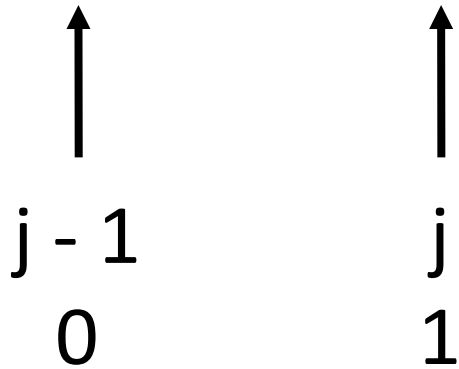
↑
j
0

Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

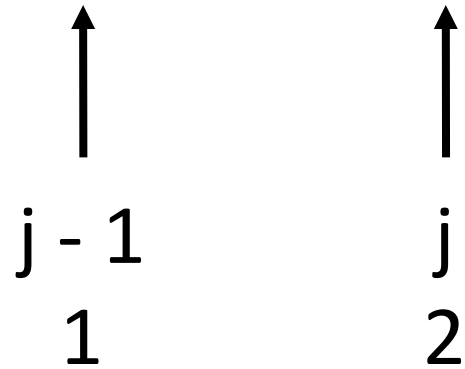


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

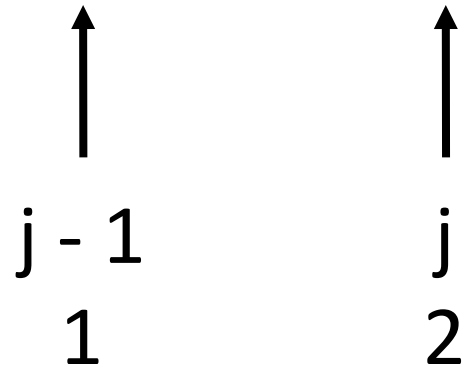
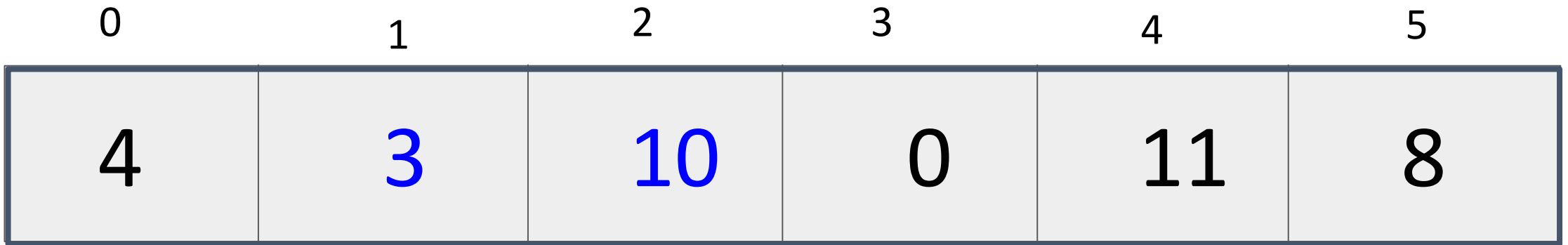


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

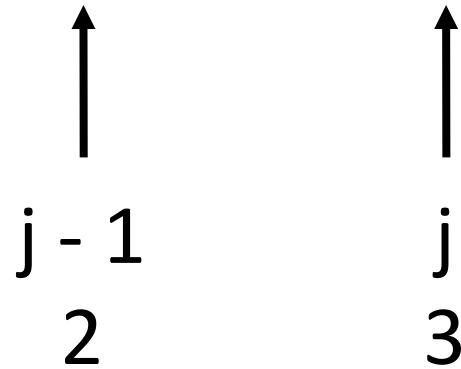


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

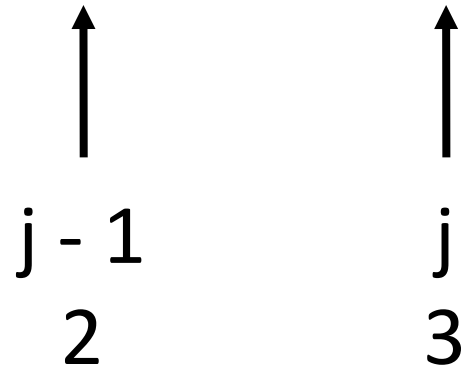
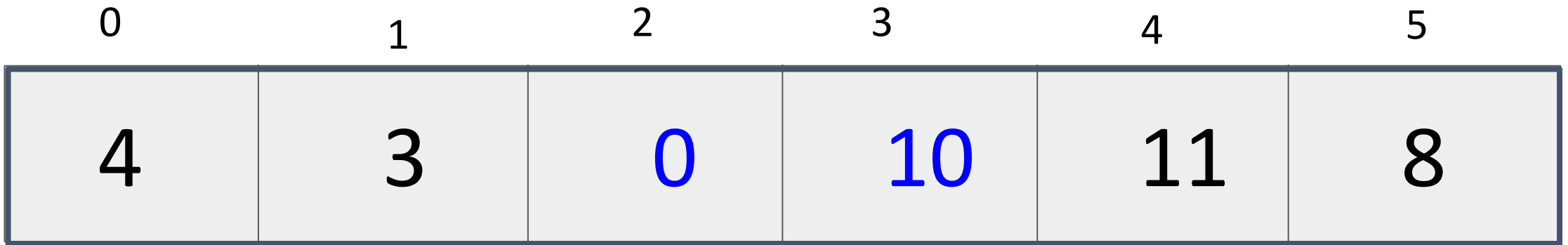


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

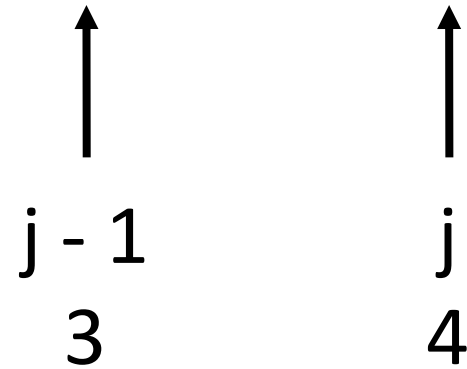
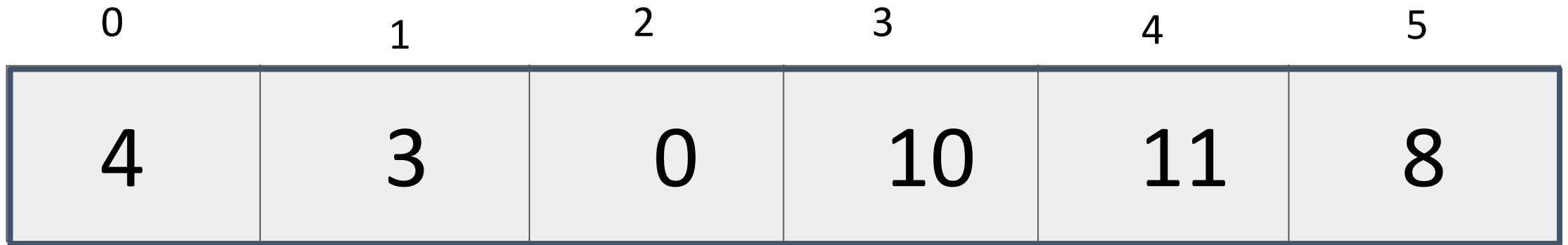


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6

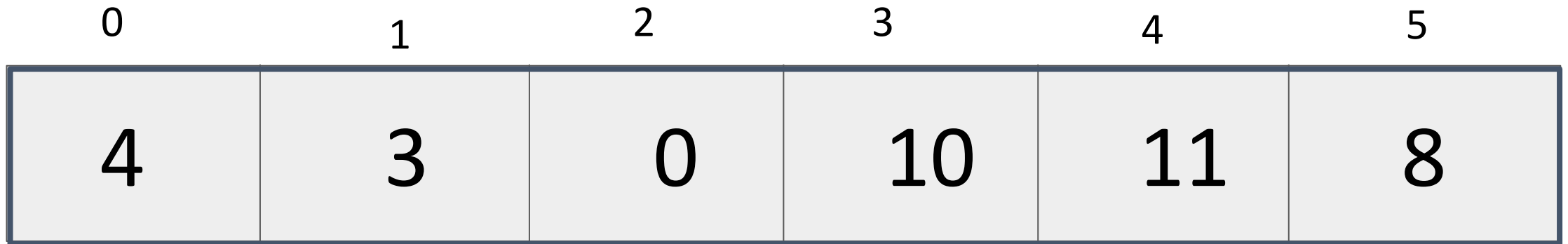


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6



↑
j - 1
4

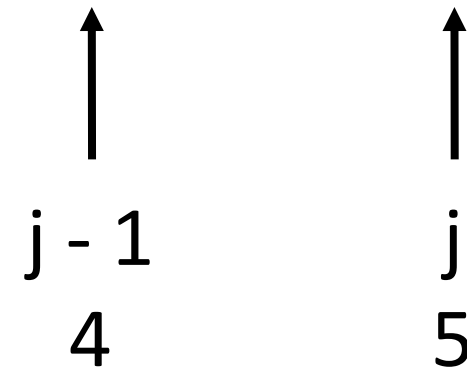
↑
j
5

Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 6



Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5



↑
j - 1
0

↑
j
1

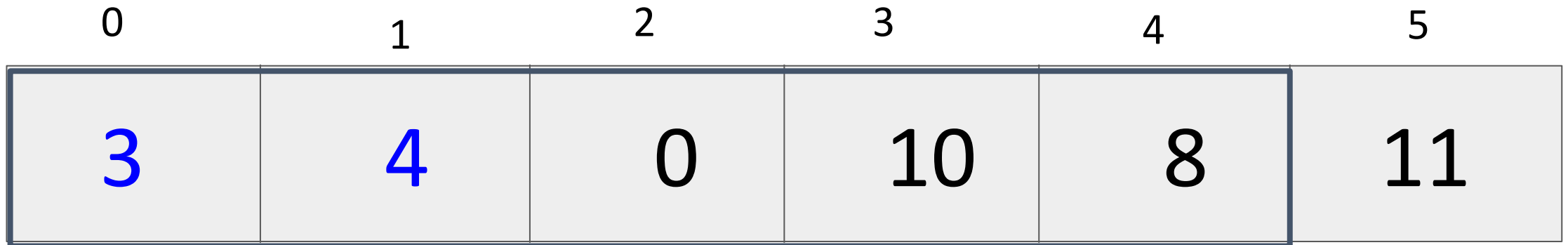
Last element has
largest element!

Reset and compare pairs with shorter list!

What next?

Bubble Sort

len = 5



↑
j - 1
0

↑
j
1

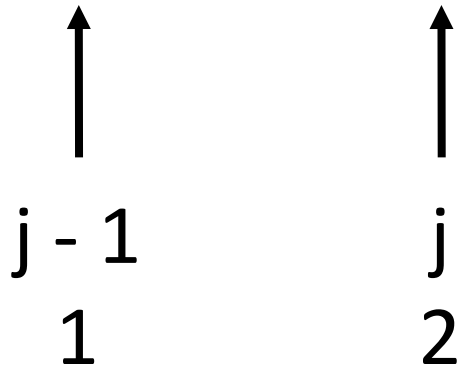
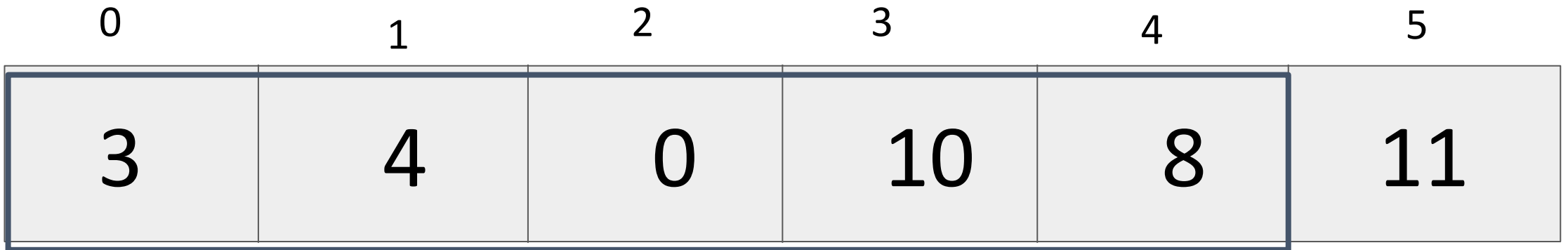
Last element has largest element!

Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5

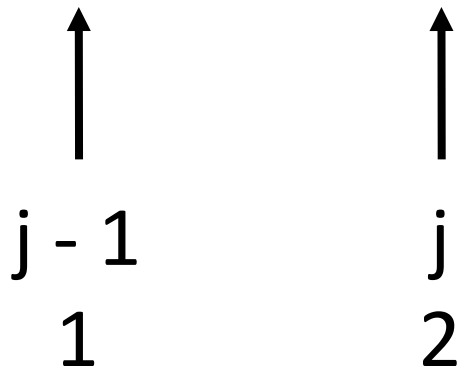
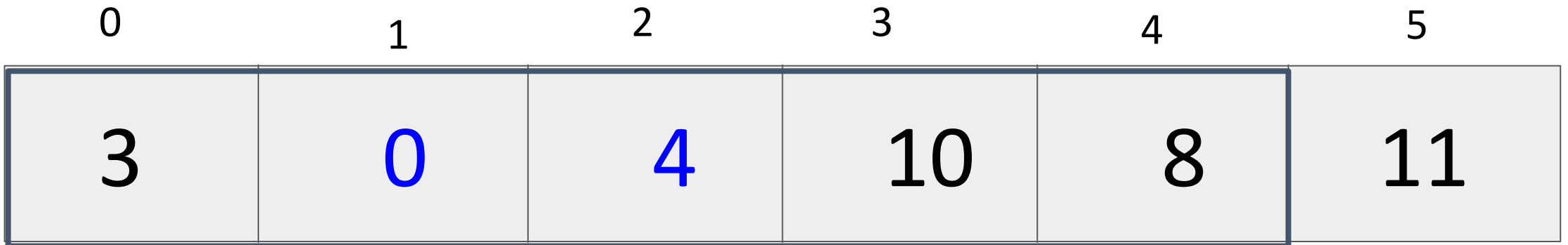


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5

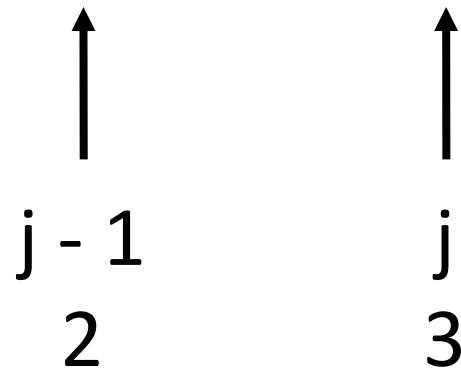
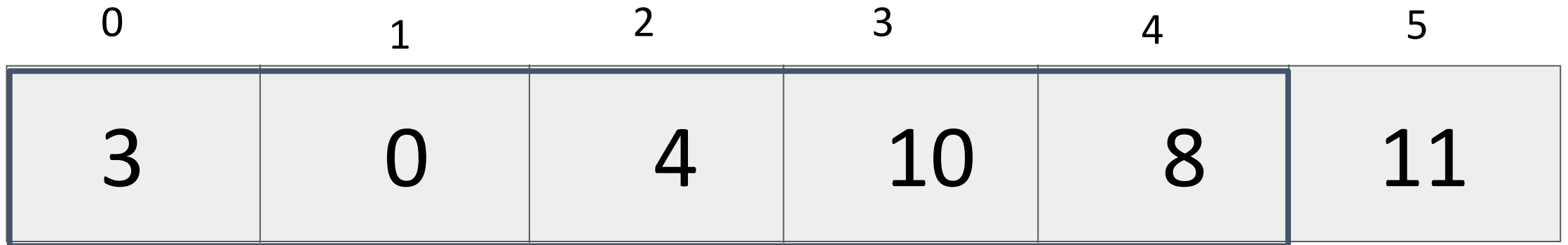


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5

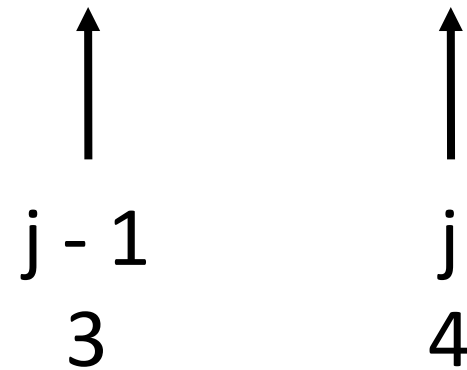
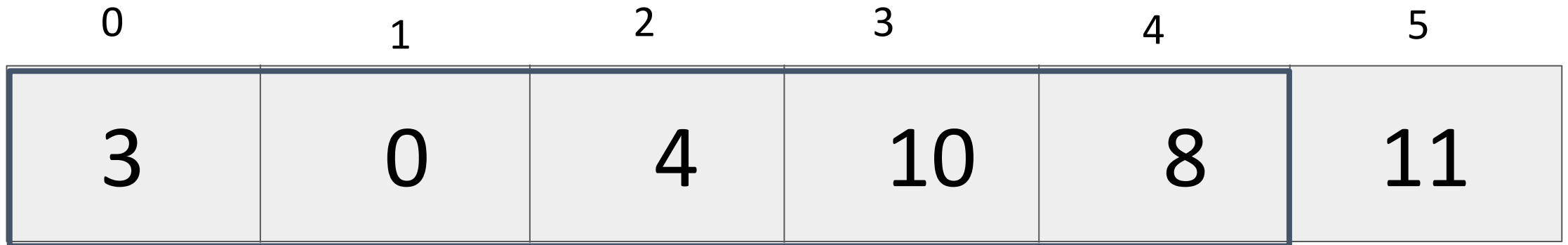


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5

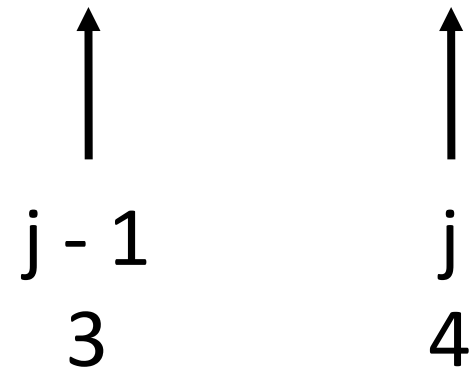
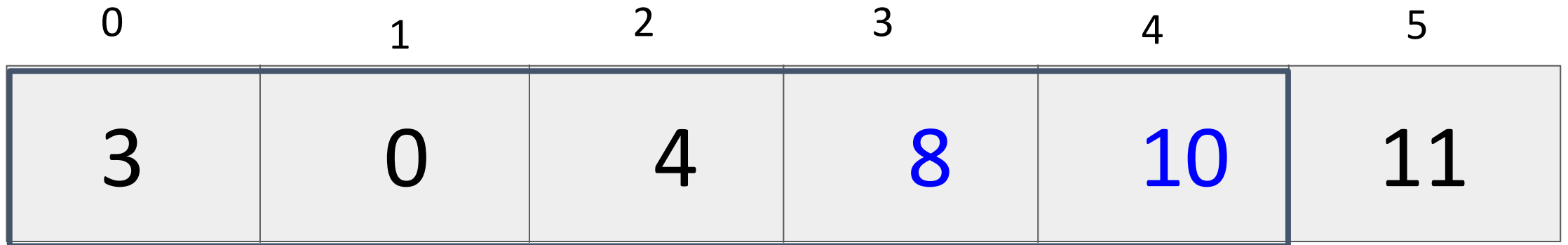


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 5

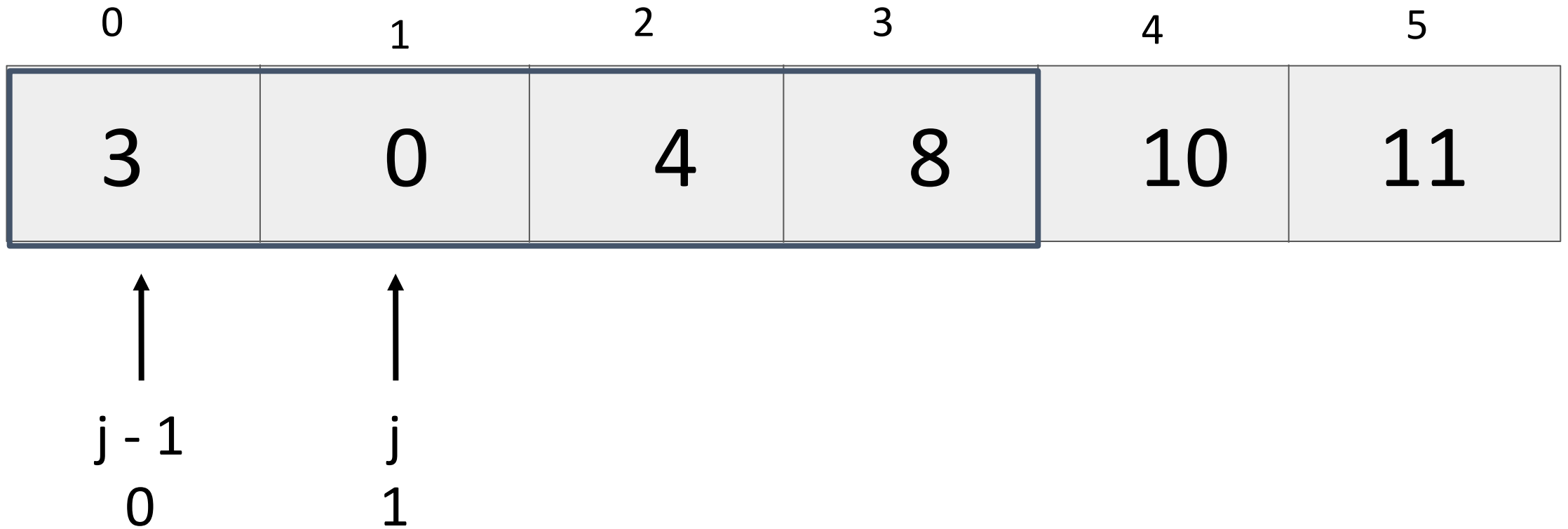


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 4

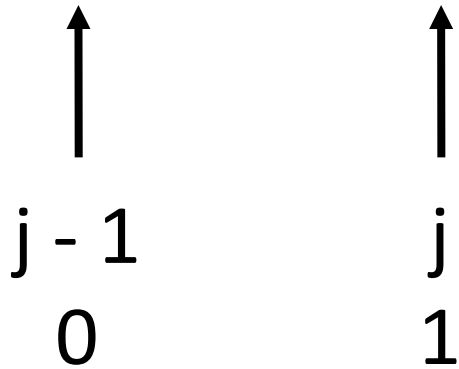
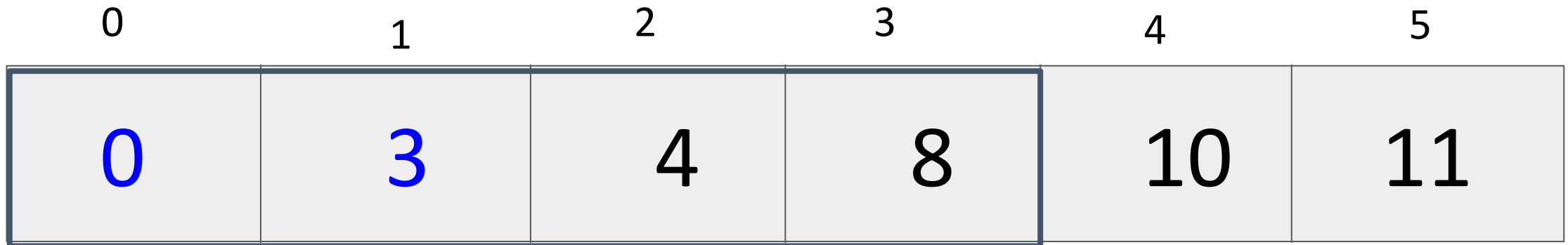


Reset and check pairs with shorter list

What next?

Bubble Sort

len = 4

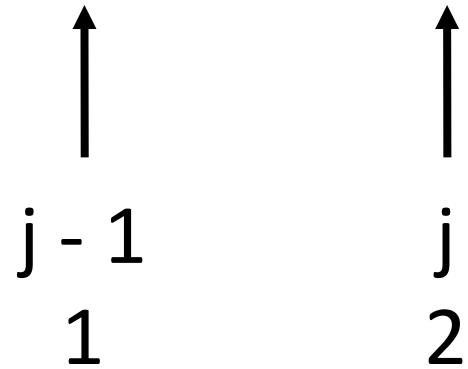
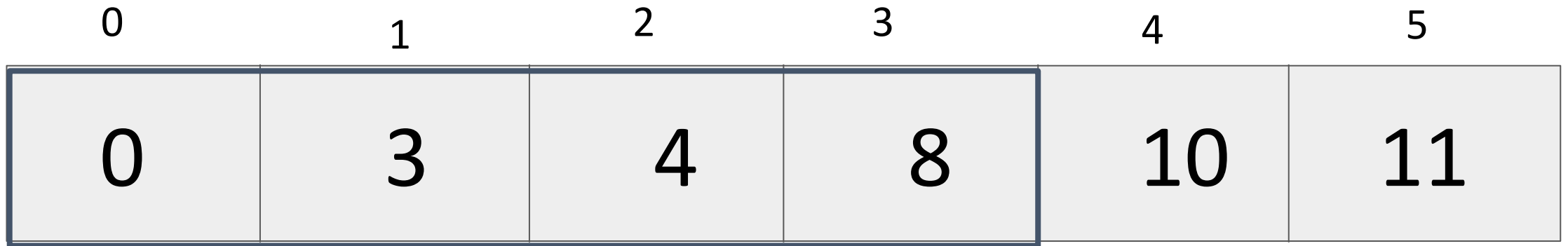


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 4

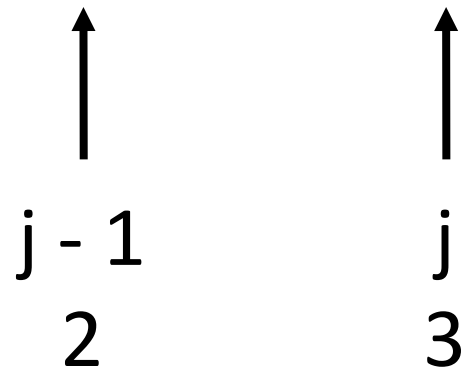
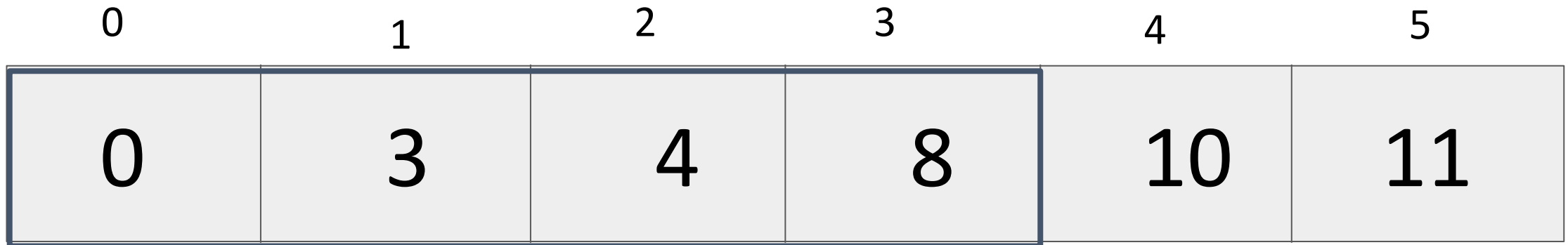


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 4

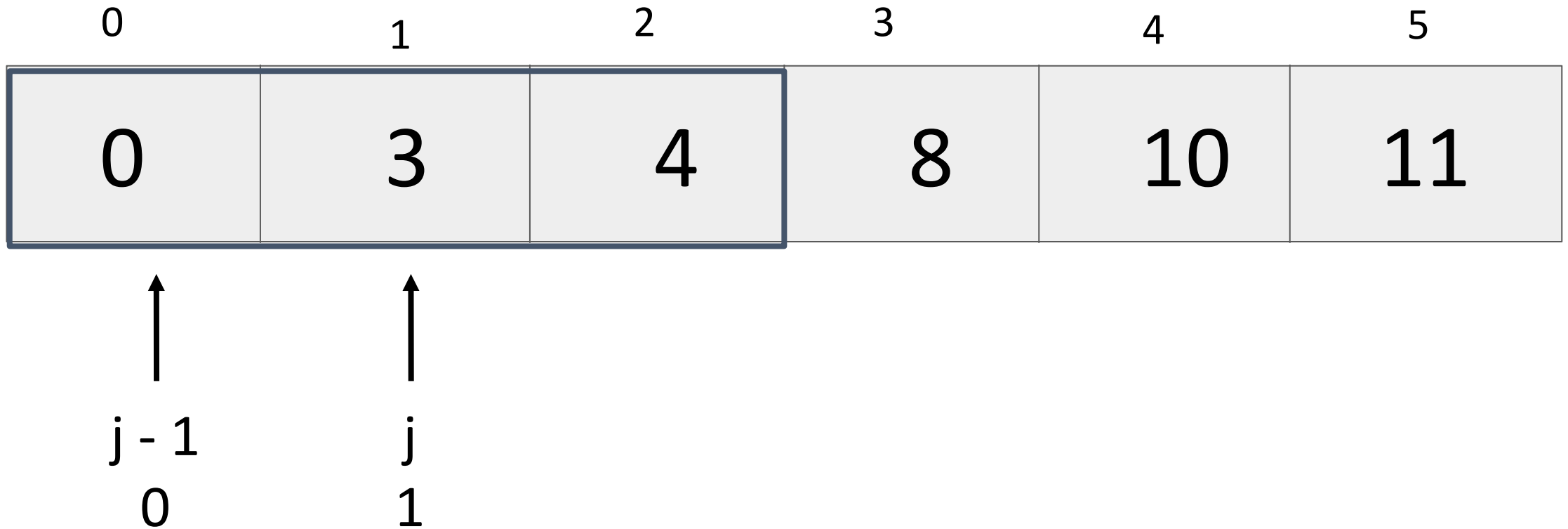


Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 3

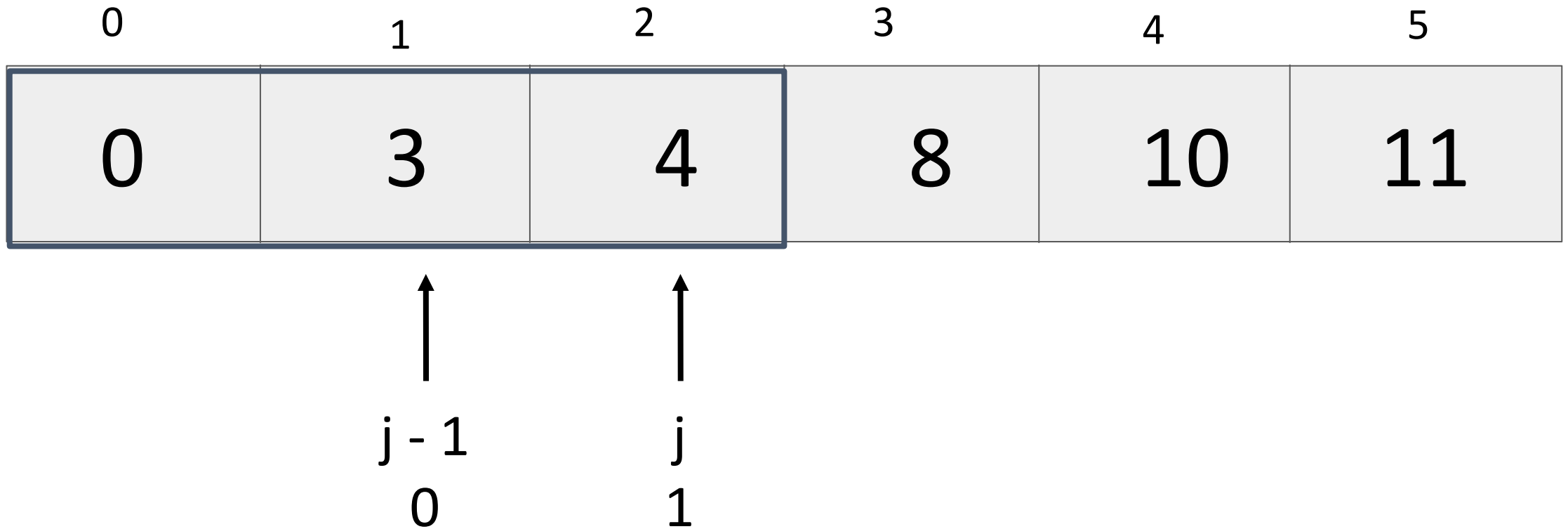


Reset; Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 3

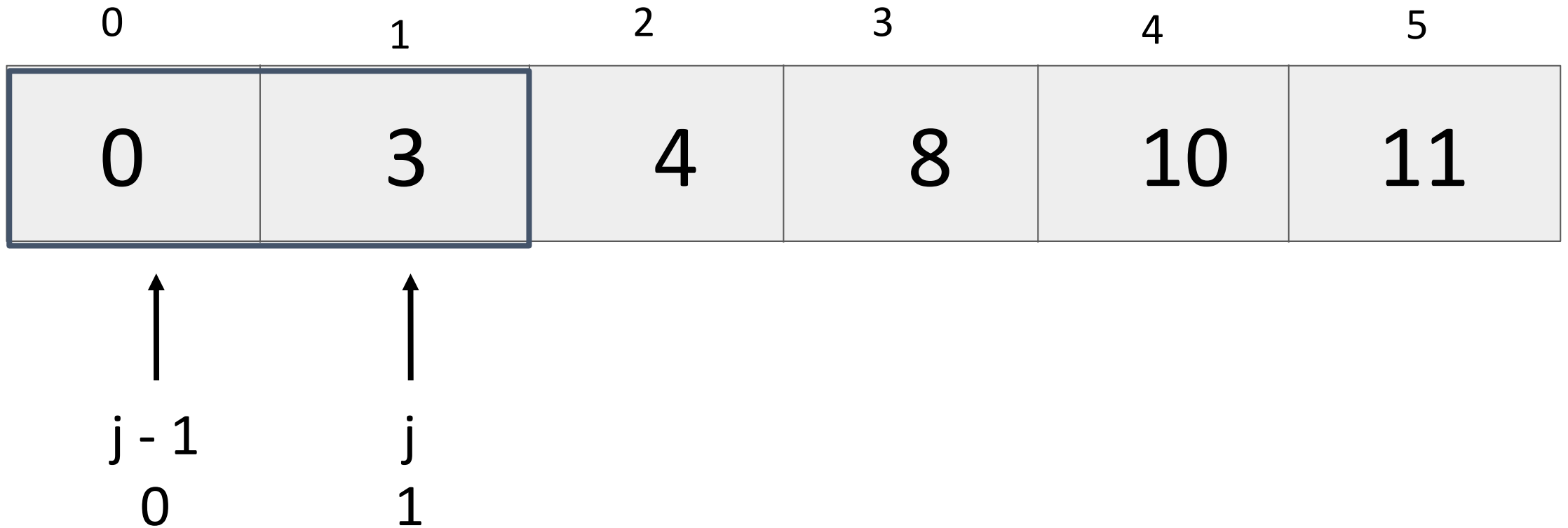


Reset; Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

len = 2



Reset; Compare $j-1$ and j ; Swap if $L[j-1] > L[j]$

What next?

Bubble Sort

Idea: bubble highest values to the end of the list; Check a shorter sublist each time

`bubbleSort(L):`

```
    for len in range(len(L), 1, -1):
```

```
        for j in range(1, len): # bubble
```

```
            if L[j-1] > L[j]:
```

```
                swap(j-1, j, L)
```

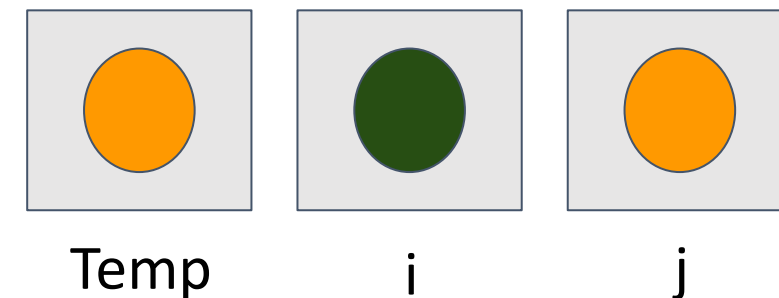
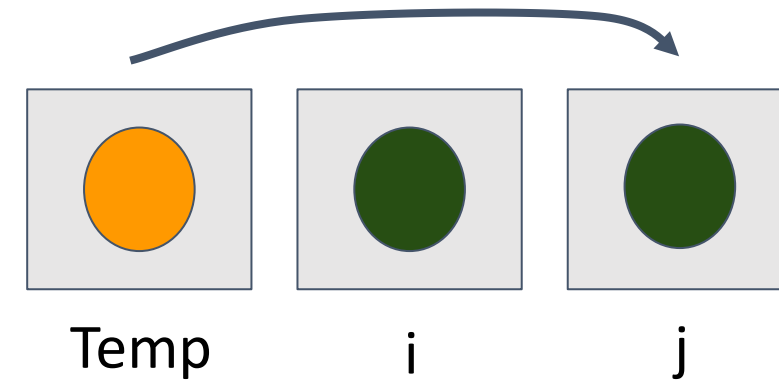
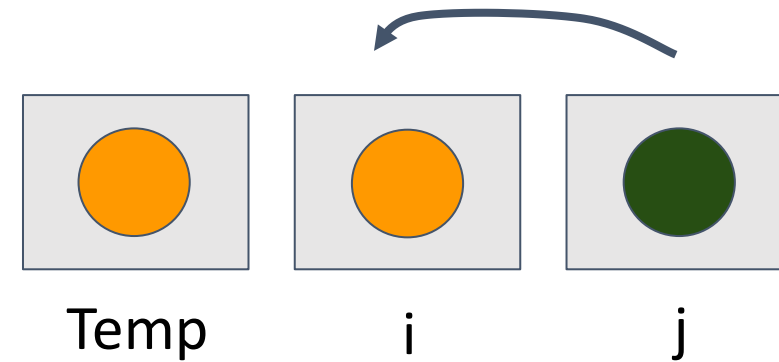
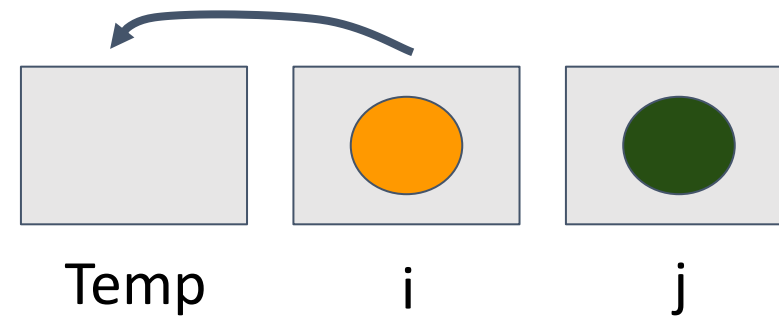
Bubble sort

swap(i, j, L):

temp = L[i] # step 1

L[i] = L[j] # step 2

L[j] = temp # step 3




```
bubbleSort.py -- ~/classes/cs21/f18/library.git/inclass/w10 -- Atom
File Edit View Selection Find Packages Help
addEx2.py useAdd.py bubbleSort.py addEx1.py
10
17 """
18
19 import swap
20 import random
21 import isSorted
22
23 def bubbleSort(L):
24     """
25     Sort the list L in place using bubble sort
26     Param L (list): the list to sort
27     Return: None
28     """
29     for end in range(len(L), 1, -1):
30         for j in range(1, end):
31             if L[j-1] > L[j]:
32                 swap.swap(j-1, j, L)
33                 print("swap", j-1, j)
34
35
36 if __name__ == '__main__':
37     L = [10,4,3,0,11,8]
38     print("Before:", L)
39     bubbleSort(L)
40     print("After:", L, "IsSorted?", isSorted.isSorted(L))
41
42
bubbleSort.py 1:1 L F N UTF-8 Python updates Fetch 21 files
```

Selection sort

Repeatedly find the smallest item and put it at front of list

selectionSort(L):

 for startIdx in range(len(L)):

 minIdx = findMinimum(startIdx, L)

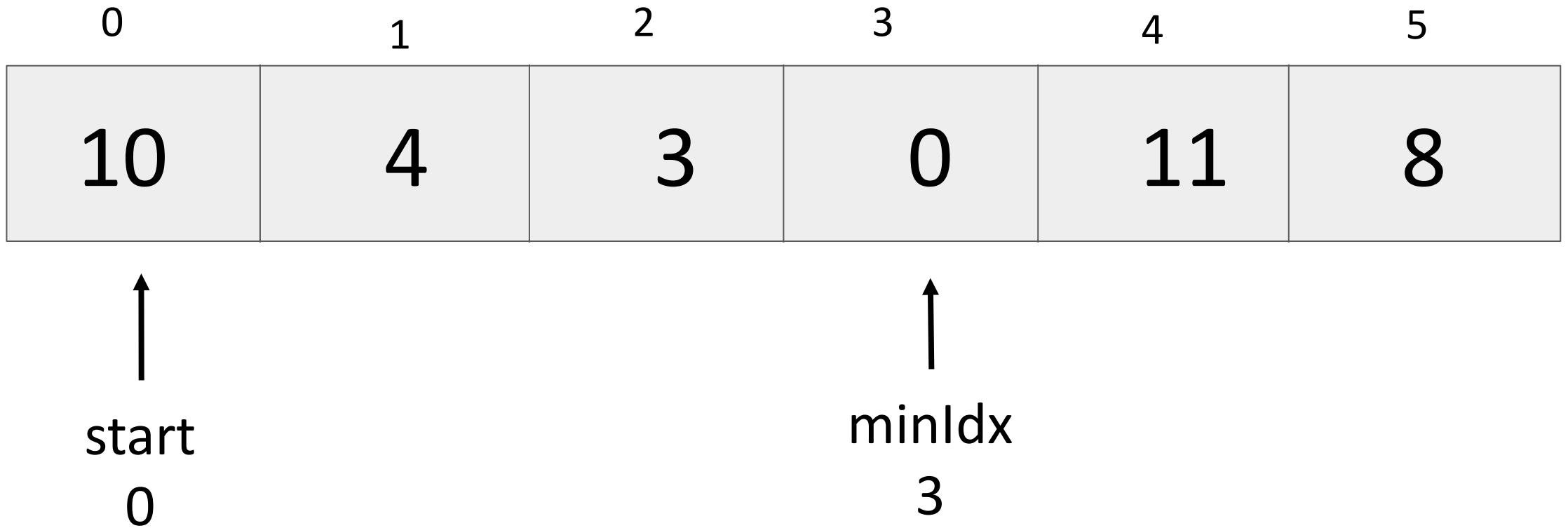
 swap(startIdx, minIdx, L)

Selection Sort

0	1	2	3	4	5
10	4	3	0	11	8

What do we do first?

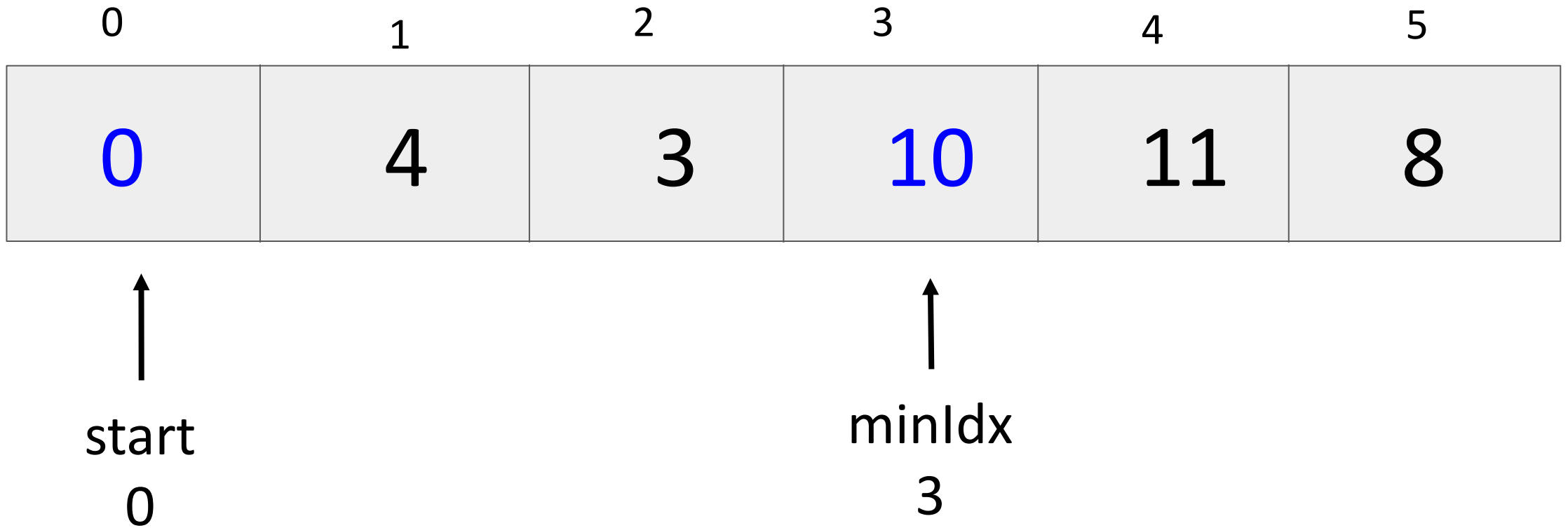
Selection Sort



Find minimum element idx between start to end

What next?

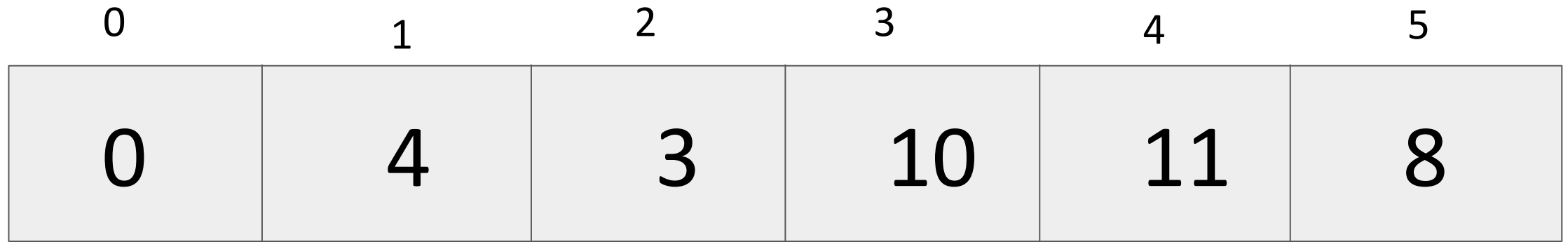
Selection Sort



Swap the elements at start and minIdx

What next?

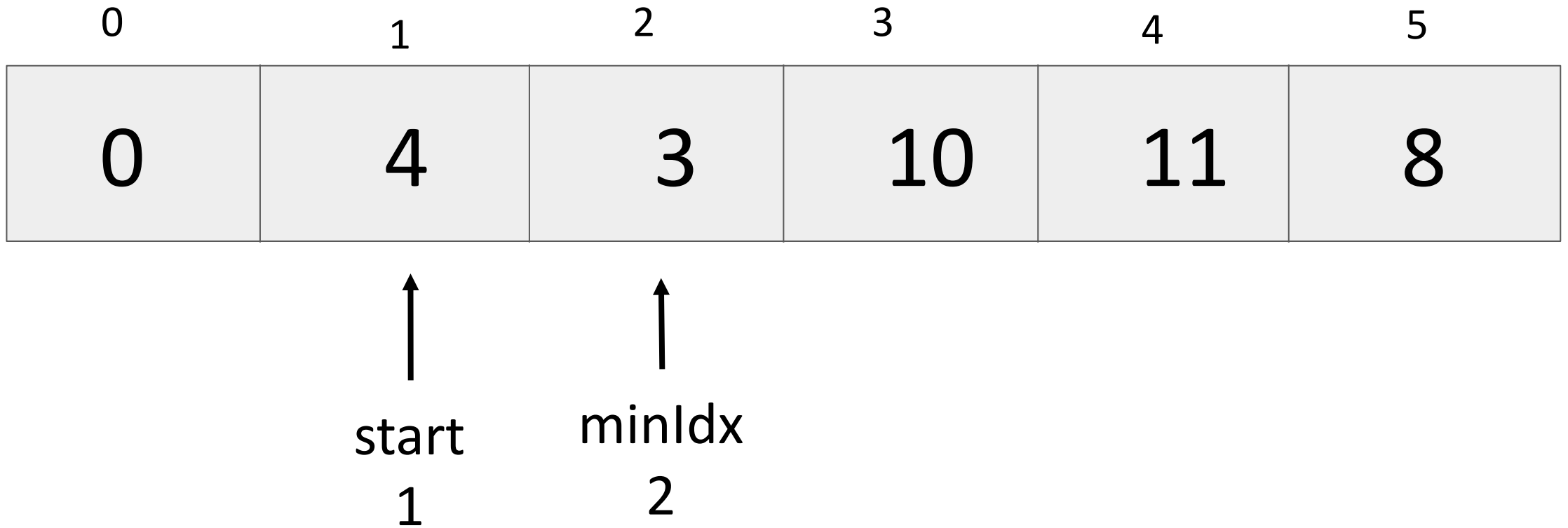
Selection Sort



Decrease the interval.

What next?

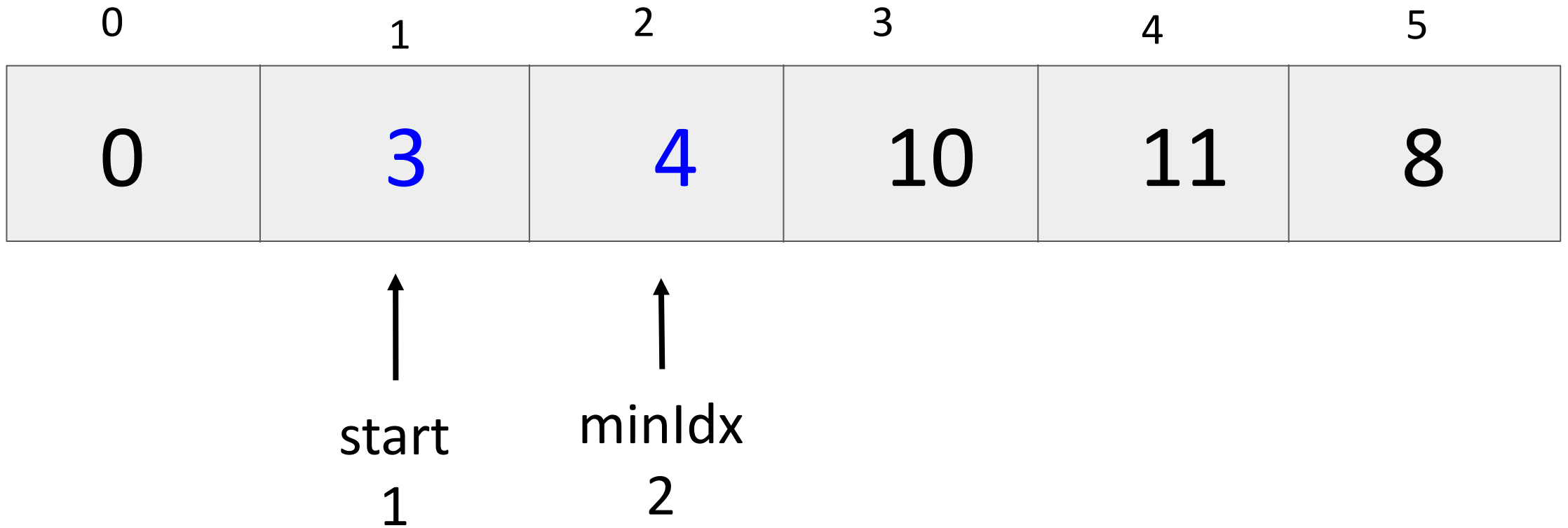
Selection Sort



Find minimum element between start to end

What next?

Selection Sort



Swap the elements at start and minIdx

What next?

Selection Sort

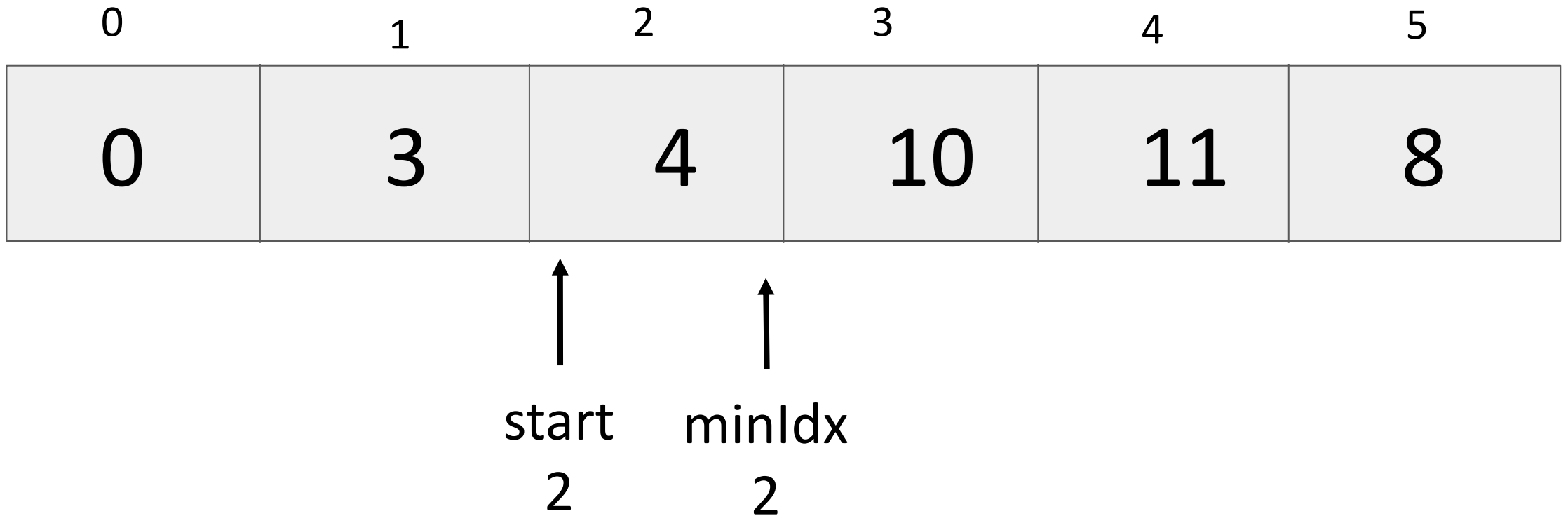
0	1	2	3	4	5
0	3	4	10	11	8

↑
start
2

Decrease the interval.

What next?

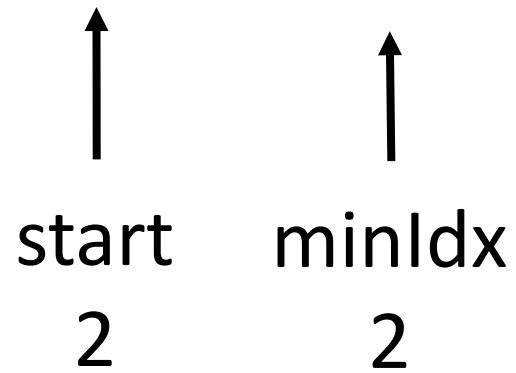
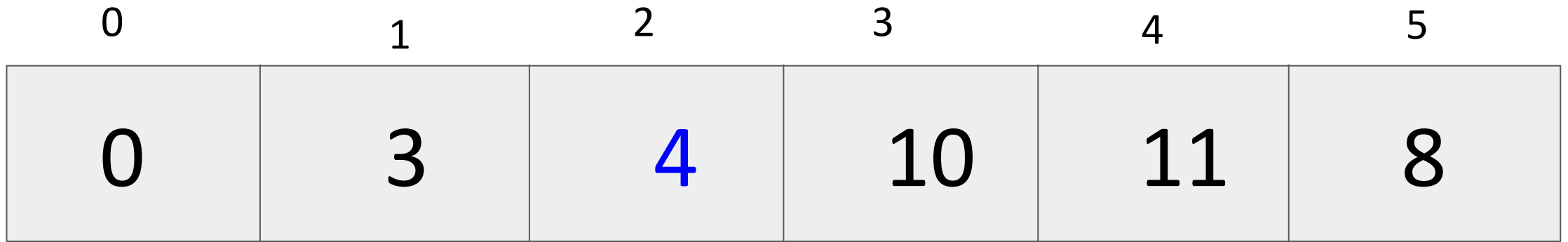
Selection Sort



Find minimum element idx between start to end

What next?

Selection Sort



Swap the elements at start and minIdx

What next?

Selection Sort

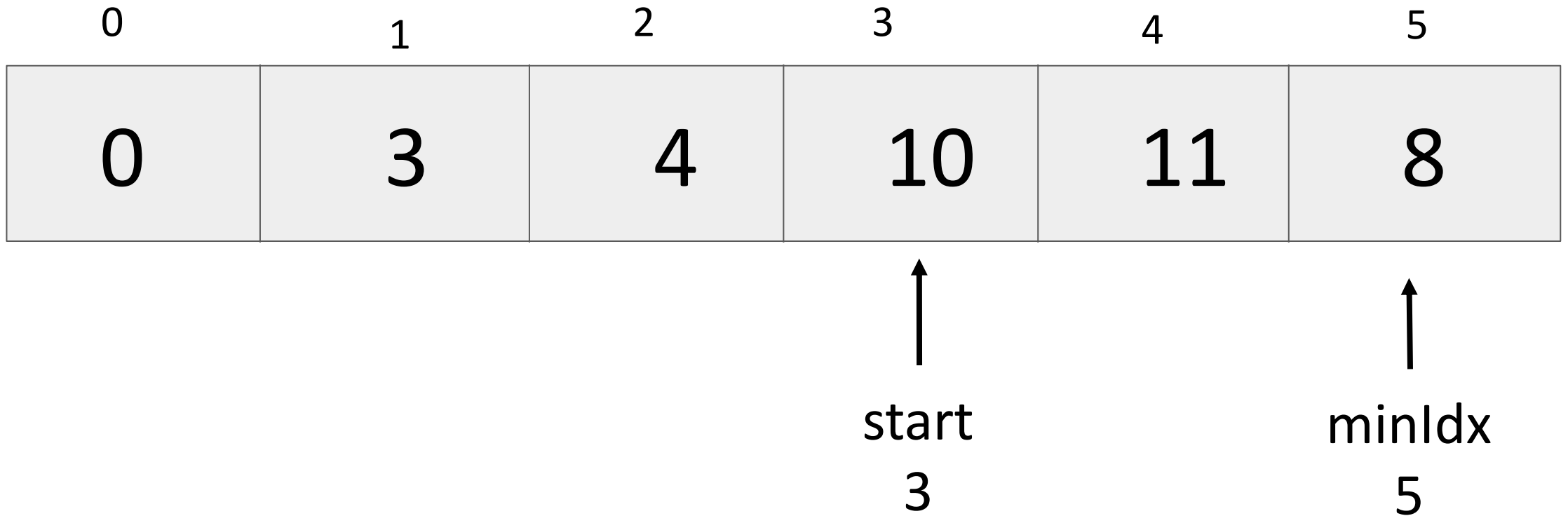
0	1	2	3	4	5
0	3	4	10	11	8

↑
start
3

Decrease the interval.

What next?

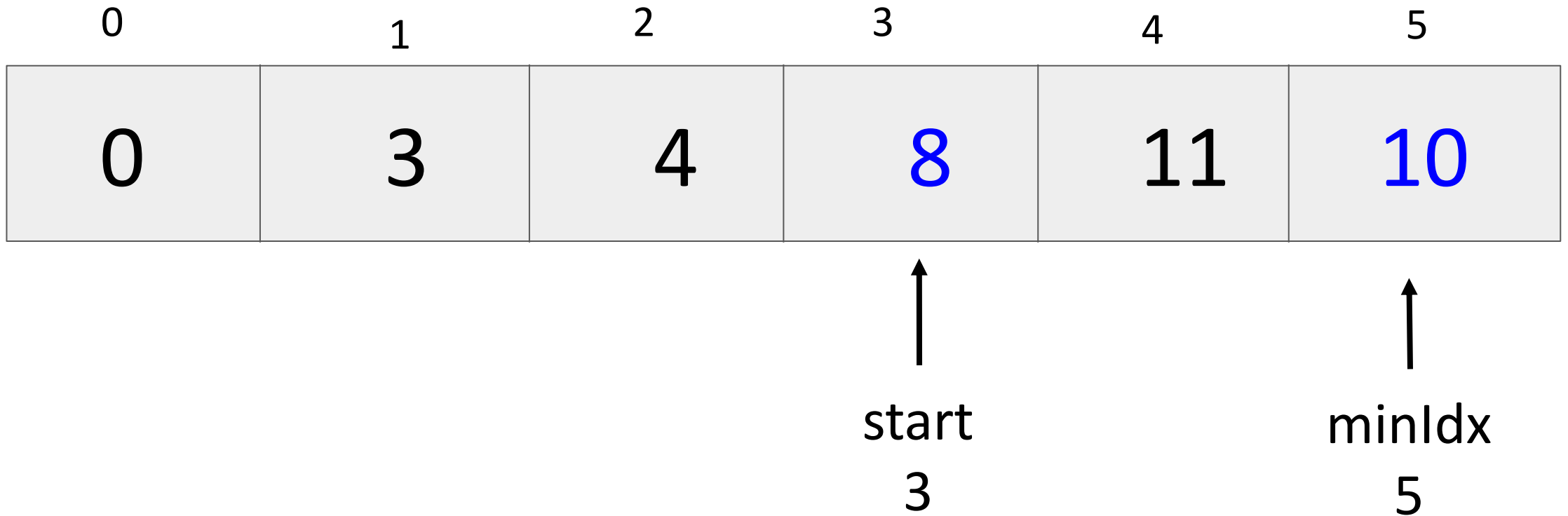
Selection Sort



Find minimum element idx between start to end

What next?

Selection Sort



Swap the elements at start and minIdx

What next?

Selection Sort

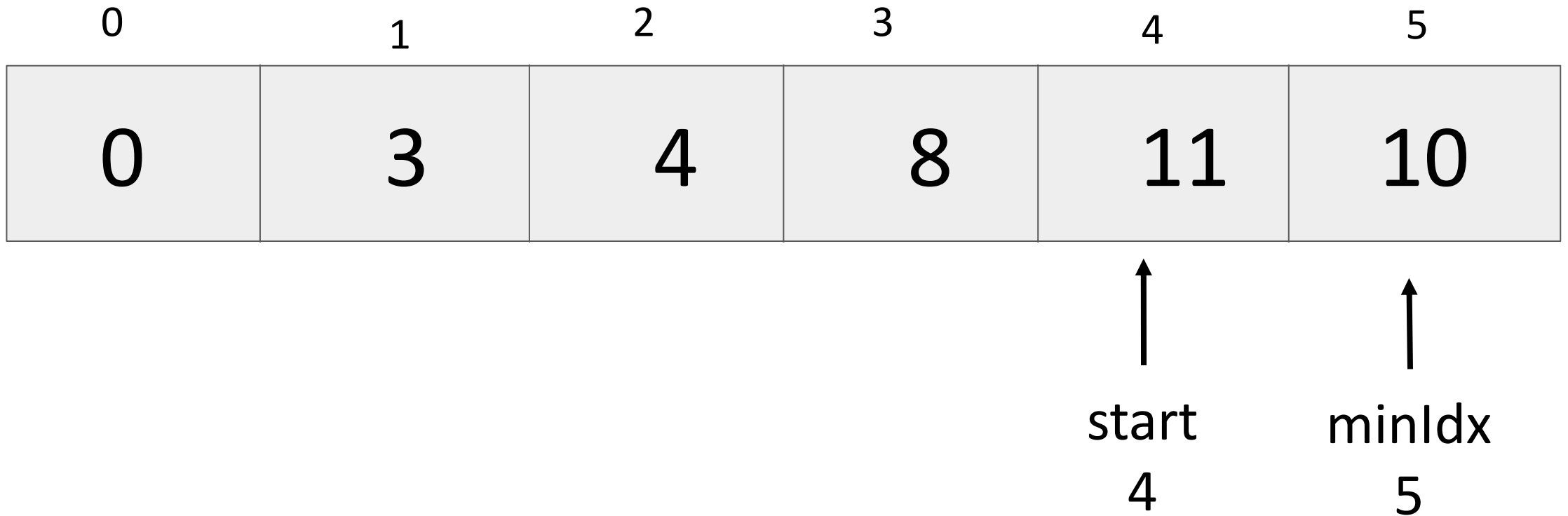
0	1	2	3	4	5
0	3	4	8	11	10

↑
start
4

Decrease the interval.

What next?

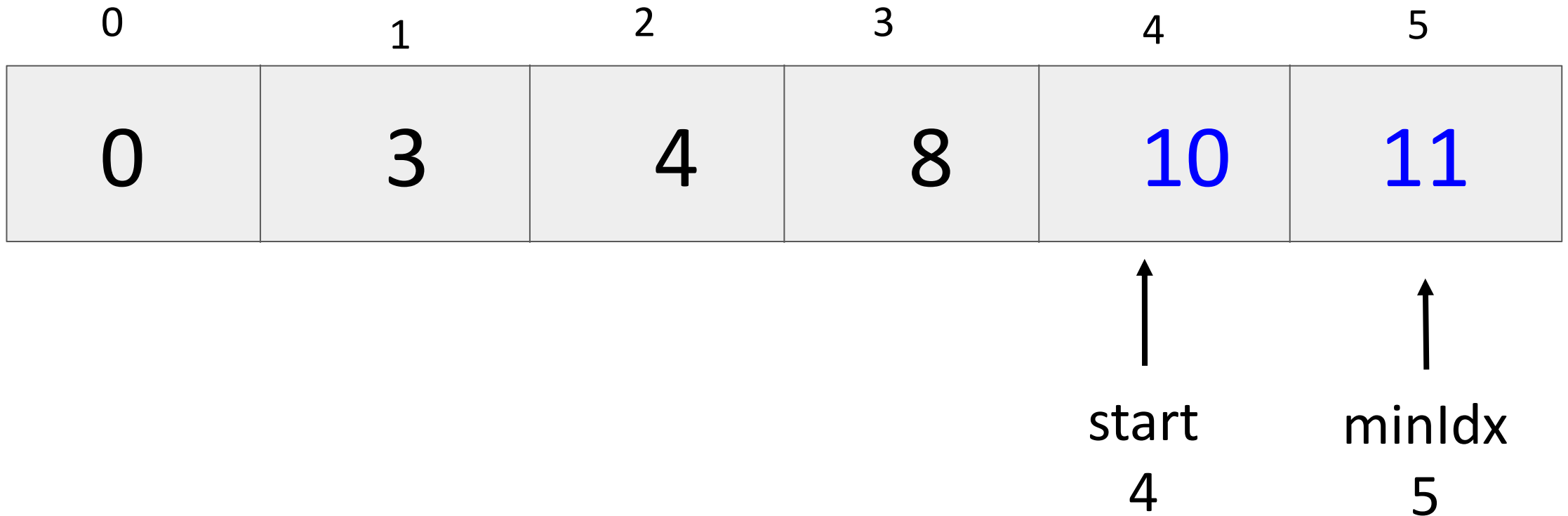
Selection Sort



Find minimum element idx between start to end

What next?

Selection Sort



Swap the elements at start and minIdx

What next?

Selection Sort



↑
start
5

Decrease the interval.

We're done!

Selection sort

```
findMinimum(startIdx, L):
```

```
    minIdx = startIdx
```

```
    for i in range(startIdx, len(L)):
```

```
        if L[i] < L[minIdx]:
```

```
            minIdx = i
```

```
    return minIdx
```

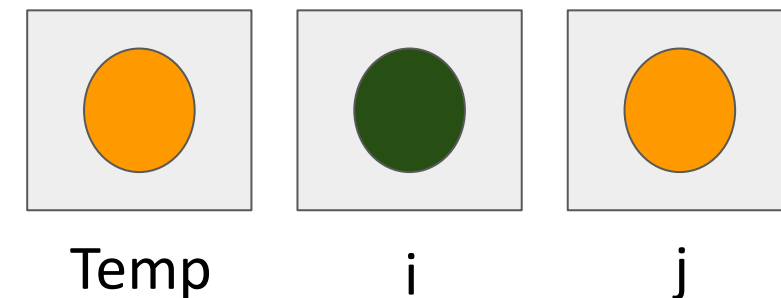
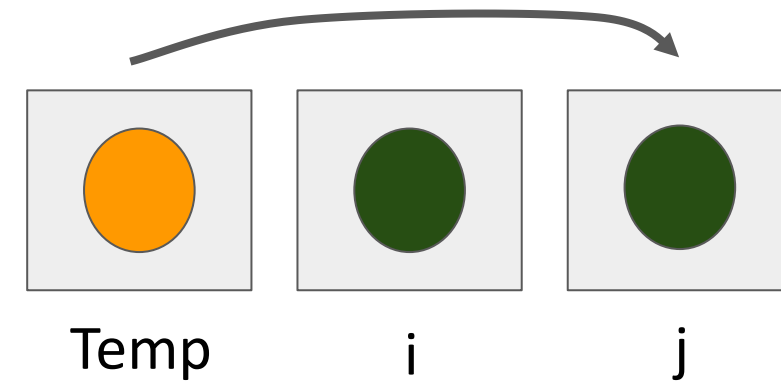
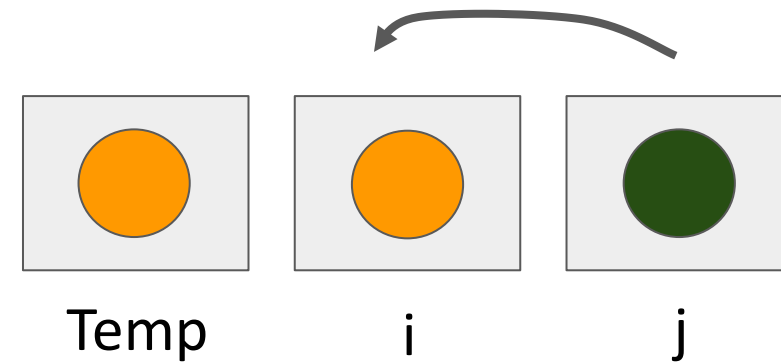
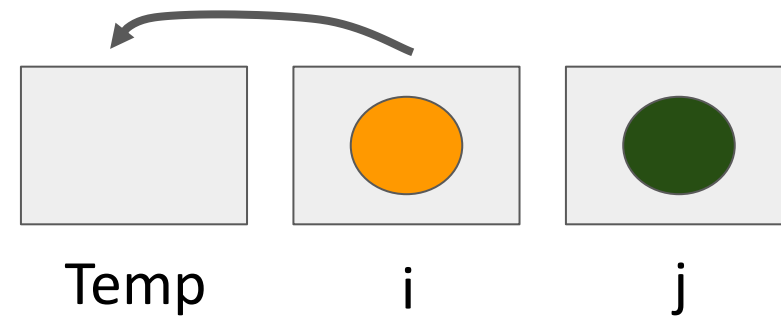
Swap

swap(i, j, L):

temp = L[i] # step 1

L[i] = L[j] # step 2

L[j] = temp # step 3



```
swap.py — ~/classes/cs21/f18/library.git/inclass/w10 — Atom
File Edit View Selection Find Packages Help
swap.py isSorted.py selectionSort.py sort-sim.py addEx2.py useAdd.py addEx1.py
1 """
2 Write a function that swaps two elements
3
4 NOTE:
5 The if statement at the bottom only executes if we run this
6 file from the command line, e.g.
7
8 >> python3 swap.py
9 Before: [0, 1]
10 After: [1, 0]
11
12 This feature allows us to use the functions in this file in other
13 programs using the syntax
14
15 import swap
16 swap.swap(0,1,L)
17
18 """
19
20 def swap(i, j, L):
21     """
22     Swaps the ith and jth elements of L
23     Params i,j (int): indexes into L
24     Param L (list): the list to change
25     Returns: None
26     """
27     tmp = L[j]
28     L[j] = L[i]
29     L[i] = tmp
30
31
32 if __name__ == '__main__':
33     L = [0,1]
34     print("Before:", L)
35     swap(0,1,L)
36     print("After:", L)
37
swap.py 1:1 LF N UTF-8 Python updates Fetch 23 files
```

```
isSorted.py — ~/classes/cs21/f18/library.git/inclass/w10 — Atom
File Edit View Selection Find Packages Help
swap.py isSorted.py selectionSort.py sort-sim.py addEx2.py useAdd.py addEx1.py
1 """
2 Write a function that checks if a list is sorted from least to greatest
3
4 $ python3 isSorted.py
5 [0, 2, 4, 6] isSorted? True
6 [6, 2, 4, 0] isSorted? False
7 [10] isSorted? True
8
9 """
10
11 import random
12
13 def isSorted(L):
14     """
15     Returns True if the list L is sorted; False otherwise
16     Param L (list): the list to test
17     Return (bool)
18     """
19     for i in range(len(L)-1):
20         if L[i] > L[i+1]:
21             return False
22     return True
23
24 if __name__ == '__main__':
25
26     L = list(range(0,8,2))
27     print(L, "isSorted?", isSorted(L))
28
29     random.shuffle(L)
30     print(L, "isSorted?", isSorted(L))
31
32     L = [10]
33     print(L, "isSorted?", isSorted(L))
34
isSorted.py 12:1 LF N UTF-8 Python updates Fetch 23 files
```

```
selectionSort.py — ~/classes/cs21/f18/library.git/inclass/w10 — Atom
File Edit View Selection Find Packages Help
swap.py isSorted.py selectionSort.py sort-sim.py addEx2.py useAdd.py addEx1.py
1 """
2 Sort a list in place using selection sort.
3 Use your existing implementation for swap and isSorted!
4
5 $ python3 selectionSort.py
6 Before: [10, 4, 3, 0, 11, 8]
7 swap 0 3
8 swap 1 2
9 swap 2 2
10 swap 3 5
11 swap 4 5
12 swap 5 5
13 After: [0, 3, 4, 8, 10, 11] IsSorted? True
14
15 """
16
17 import swap
18 import random
19 import isSorted
20
21 def FindMinimumIdx(start, L):
22     minVal = L[start]
23     minIdx = start
24     for i in range(start, len(L)):
25         if L[i] < minVal:
26             minVal = L[i]
27             minIdx = i
28
29     return minIdx
30
31 def selectionSort(L):
32     """
33     Sort the list L in place using selection sort
34     Param L (list): the list to sort
35     Return: None
36     """
37     for start in range(len(L)):
38         minVal = L[start]
39         minIdx = start
40         for i in range(start, len(L)):
41             if L[i] < minVal:
42                 minVal = L[i]
43                 minIdx = i
44             swap.swap(minIdx, start, L)
45             print("swap", start, minIdx)
46
47 if __name__ == '__main__':
48     L = [10,4,3,0,11,8]
49     print("Before:", L)
50     selectionSort(L)
51     print("After:", L, "IsSorted?", isSorted.isSorted(L))
52
selectionSort.py 2:27 LF N UTF-8 Python updates Fetch 23 files
```

Selection sort and Bubble sort are $O(N^2)$

